[[1]](#footnote-0)

Preparation of Papers for IEEE TRANSACTIONS and JOURNALS(December 2013)

First A. Author, *Fellow, IEEE*, Second B. Author, and Third C. Author, Jr., Member, IEEE

*Abstract*—These instructions give you guidelines for preparing papers for IEEE Transactions and Journals*.* Use this document as a template if you are using Microsoft *Word* 6.0 or later. Otherwise, use this document as an instruction set. The electronic file of your paper will be formatted further at IEEE. Paper titles should be written in uppercase and lowercase letters, not all uppercase. Avoid writing long formulas with subscripts in the title; short formulas that identify the elements are fine (e.g., "Nd–Fe–B"). Do not write “(Invited)” in the title. Full names of authors are preferred in the author field, but are not required. Put a space between authors’ initials. Define all symbols used in the abstract. Do not cite references in the abstract. Do not delete the blank line immediately above the abstract; it sets the footnote at the bottom of this column.

*Index Terms*—Enter key words or phrases in alphabetical order, separated by commas. For a list of suggested keywords, send a blank e-mail to [keywords@ieee.org](mailto:keywords@ieee.org) or visit <http://www.ieee.org/organizations/pubs/ani_prod/keywrd98.txt>

# INTRODUCTION

A

T present, more and more people, especially youth, like to use LBSN(Location-based Social Network), which is a kind of social network. Unlike the traditional social network(ex. facebook), besides contacting others by network, an individual also can share the location information to others.

Foursquare is a typical LBSN app.It collects user’s browsing , purchasing, and check-in activities history and provides local search-and-discovery service. Based on Foursquare’s database, users can easily get personalized recommendations of places to go to near his/her current location.

In this document, we will use some db tools and part of the real dataset of Foursquare

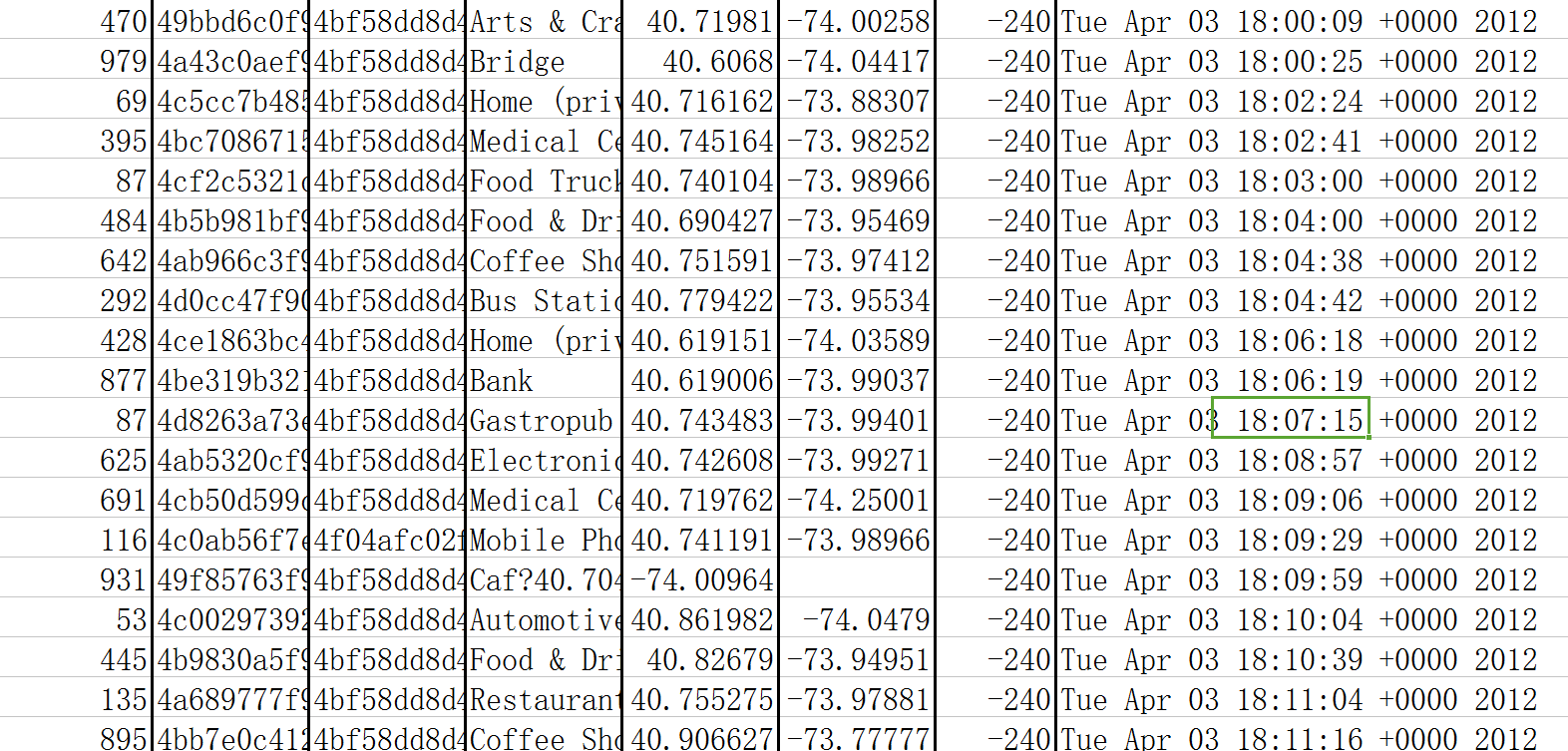
to do some simple and complicated queries to get a series of users and venues ranking list.For instance:Data can be used to study the popularity of venues in many categories based on user check-ins and hence help to make predictions and recommendations to other users, investors, venue holders , or even taxi drivers. The data is mostly provided by highly active social media users & therefore the recommendations created will also be most useful for this audience.

# REQUIREMENT ANALYSIS

We select the NYC Check-in Dataset, this dataset contains check-ins in NYC collected for about 10 month (from 12 April 2012 to 16 February 2013). It contains

227,428 check-ins in New York city. Each check-in is associated with its time stamp, its GPS coordinates and its semantic meaning (represented by fine-grained venue-categories). This dataset is originally used for studying the spatial-temporal regularity of user activity in LBSNs. The real-time updating & processing of this data requires a system capable of handling a high number of Read/Write operations at speed.

Raw data:



This dataset includes long-term (about 10 months) check-in data in New York city collected from Foursquare from 12 April 2012 to 16 February 2013.

From left to right, the 8 columns are:

1. User ID (anonymized)

2. Venue ID (Foursquare)

3. Venue category ID (Foursquare)

4. Venue category name (Fousquare)

5. Latitude

6. Longitude

7. Timezone offset in minutes (The offset in minutes between when this check-in occurred and the same time in UTC)

8. UTC time

The requirements(data analysis questions) consist of 5 queries:3 complex and 2 simple queries:

|  |
| --- |
| SIMPLE:  Summarising - Most popular places?  Summarising - Most active users?  COMPLEX:  How many other venues of the same category are nearby?  Most popular places of the most active users of each category?  Geo-spatial Distribution over Time? |

We will handle them separately:

Simple question solution

Q1 Most popular places? Obviously, “most popular” means having most visiters, in this dataset, it means a venue contained in most check-in lines. Algorithm is simple: count all Check Ins for each venue, Display Highest value;

Q2 Most active users? Like the 1st query, “most active” means having most check-in times, so we have to find the user contained in most check-in lines. Algorithm is also quite

simple:Count all Check Ins by each User, Display Highest value;

Complex question solution

Q3 How many other venues of the same category are nearby?

Given a certain venue, “nearby” can be defined by max distance value, only the venues have the less value can be seen as “nearby”. So the algorithm is to map all venues of each

category. For each venue, calculate distances apart,Count all venues within proscribed radius

Q4

Most Popular places of the most active users of each category?

For Each Category Count Check Ins by each ??????????

User

Display most Checked into venue for the MAU.

Q5 Geo-spatial Distribution over Time?

Map Check Ins

from the last hour to a grid

Count check Ins for each Grid reference cell

?????????????

All these queries can be transformed to pure statistics question, so it’ll be possible to implement the solution by some db tools and software program.

# Evaluation

## testing environment

First we list some parameters of our testing environment:

Data set: as shown in part II, the whole dataset is 30,743KB, including 227428 records

Computer hardware:

CPU: Intel i5-2450M 2.50GHz

RAM:RAM 4GB

Operating system: Windows 10 - 64 bit

Software versions:

Java Runtime Environment(JRE) : 1.8.0\_121

Java IDE: Eclipse Jee Neon

Redis:

Driver: jedis-2.8.1.jar

Redis 3.2.100 64 bit

Cassanddra:

Driver: cassandra-driver-core-3.1.0.jar

Cassandra 3.0.9

CQL spec 3.4.0

cqlsh 5.0.1

## Queries test result and performance analysis

In the 1st step, we create both the dataset for Redis and Cassandra respectively:

Totally 227428 records

|  |  |  |
| --- | --- | --- |
|  | **Cassandra** | **Redis** |
| Upload time cost(ms) | 6104416 | 99803 |
| Performance(records/s) | 37.25630756 | 2278.769175 |

Cassandra's poor performance in this case can be due to its columner database structure. In most tables in our data model, we only have to query within 2-3 attributes, in Redis, we can

build simple dataset like hmset, zset, while in Cassandra, we have to build every data model with a column based table, which will definitely cost more resources.

In the 2nd step, we handle the queries one by one:

For Q1-Q5, the result and performance are as follows:

|  |  |  |
| --- | --- | --- |
|  | **Cassandra** | **Redis** |
| Q1 time cost(ms) | 129477 | 2110 |
| Q2 time cost(ms) | 7551 | 98 |
| Q3 time cost(ms) | 3305 | 84 |
| Q4 time cost(ms) |  |  |
| Q5 time cost(ms) | 2344 | 55 |

|  |  |  |
| --- | --- | --- |
|  | **Cassandra** | **Redis** |
| Q1 results |  |  |
| Q2 results |  |  |
| Q3 results |  |  |
| Q4 results |  |  |
| Q5 results |  |  |

In this step , Cassandra also has a poor performance, this is attributed to both data structure and methods.

Our queries keep focus on counting times, in Redis, the dataset has been built by ID sets and has easy and direct method to get a count result. For example: for different venues, we

have zsets with keys ranging from venueID1 to venueIDn. After we built up the Redis dataset, for the venueIDi (0<i<n), all the relevant check-in for the reference venue items has been added in zset(“venueIDi”). Then we can easily get the total check-in count by invoking a zcount method on zset with the key of “venueIDi”.And even in java, we can call jedis.zcount() directly to get its member count value.

However, it is difficult to use columns to create a dataset like zset in Redis , and in Cassandra CQL, there is no direct method for counting group by members, we have to generate the count statement first and then do the query, this process would cause a low efficiency.

？？？？？？？Caculate distance

Another important reason is, Redis has disk-backed in memory database, which enable Redis work faster than Cassandra.

# Conclusion and Future development

## conclusion

In this case, the data attributes number is small and data structure is simple, so Redis is more suitable for it as it has a variety of simple dataset types with helpful interface.

1. This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported in part by the U.S. Depart­ment of Com­merce under Grant BS123456”.

   The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

   S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

   T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp). [↑](#footnote-ref-0)